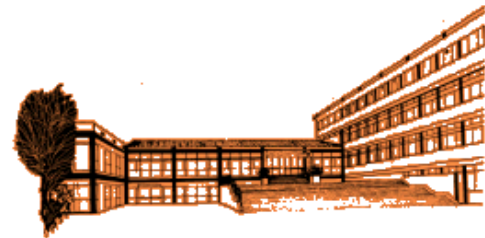




# ISEL

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA



## Instituto Superior de Engenharia de Lisboa



Departamento de Engenharia da Electrónica das  
Telecomunicações e dos Computadores








# Sistemas Computacionais Distribuídos

WEB - Geração de Páginas dinamicamente - CGI's

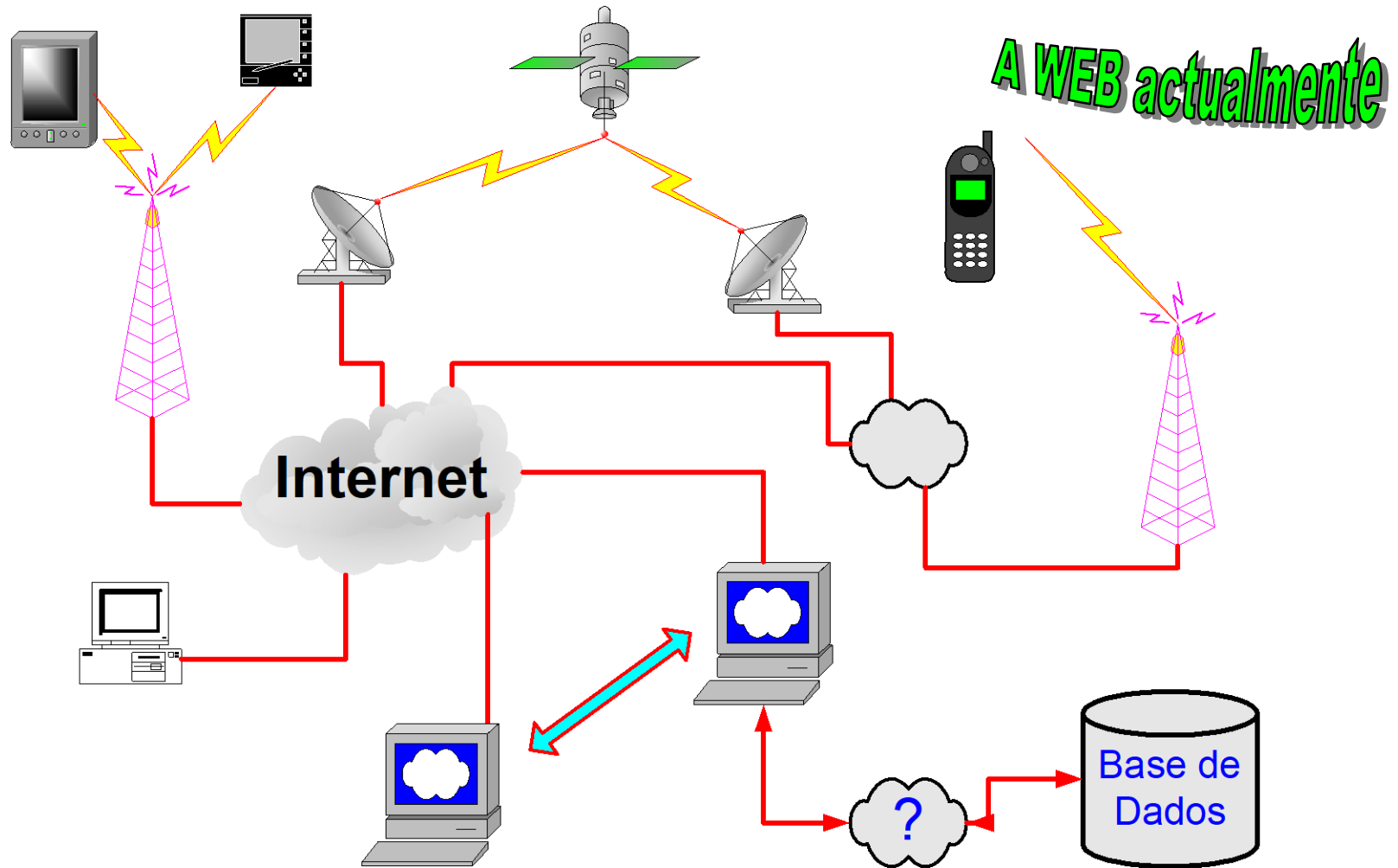


# Índice

-  Introdução
-  Necessidade de Gerar Páginas Dinamicamente
-  O que é um CGI?
-  Passagem de dados (servidor web ↔ CGI)
-  Exemplos



# Introdução





# Introdução

- A WEB oferece um mecanismo simples para publicação de informação estática;
  - Através da criação de páginas HTML que residem no servidor WEB
- As páginas estáticas não permitem interacção contextualizada com o utilizador;
- Existe a necessidade de poder interagir com o servidor WEB de modo a ser possível realizar operações interactivas;
  - Ex: Consulta de email, Banca online, e-commerce, etc...



# Necessidade de Gerar Páginas Dinamicamente

- ❏ O protocolo HTTP oferece de raiz mecanismos para transferir para o servidor os dados que o utilizador gera;
  - ❏ Métodos GET, POST, ...;
- ❏ Os servidores WEB têm de processar, dinamicamente, os pedidos efectuados pelos clientes;
- ❏ Que fazer com esses dados?





# Necessidade de Gerar Páginas Dinamicamente

- ❏ Os servidores WEB não sabem, de um modo geral, o que fazer com os dados dinâmicos que recebem;
- ❏ É “*impossível*” ter um servidor que saiba processar todos os dados gerados dinamicamente;



# Necessidade de Gerar Páginas Dinamicamente

- ❏ Em alternativa os servidores disponibilizam ferramentas para processar os dados;
- ❏ O método tradicional mais utilizado é usar o mecanismo *gateway programs*.



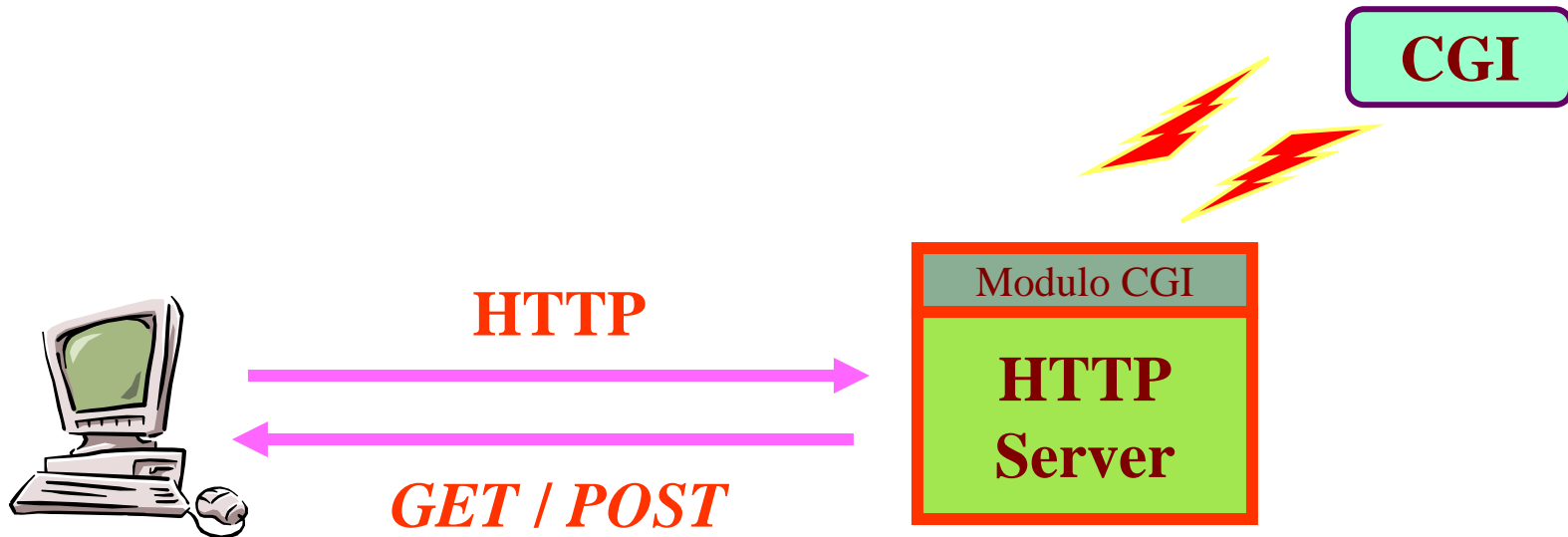
# CGI – *Common Gateway Interface*

- ❏ Foi o primeiro mecanismo a surgir que possibilitava a geração dinâmica de informação na WWW;
- ❏ As páginas podem assim ser construídas em tempo real em vez de terem que ser previamente guardadas em ficheiros no servidor;
- ❏ Consiste na execução de um programa no servidor de WWW desencadeado pelo acesso, de um utilizador, a uma determinada página;











# O Conceito de *gateway programs*



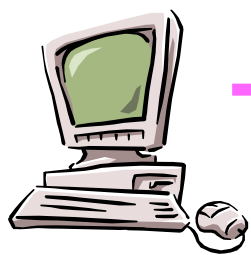


# CGI – Modo de funcionamento

-  O servidor WEB lança uma aplicação CGI se for feito um pedido HTTP que a refira;
-  Antes de o servidor WEB lançar o CGI define um conjunto de variáveis de ambiente;
  -  Ex: QUERY\_STRING, REQUEST\_METHOD, etc...
-  Redirecciona o <stdin> e <stdout> de modo ao CGI interagir por omissão com o servidor web;
-  O programa CGI lê os dados à sua disposição, interpreta esses dados e envia como resposta uma página HTML;
-  O servidor web recebe a página de resposta do CGI e devolve-a na mensagem HTTP de resposta ao pedido feito originalmente pelo browser;



# O Conceito de *gateway programs*



1



**HTTP  
Server**

1. O Cliente envia os dados para o servidor:

- ⌚ A *query string* como parte do URL;

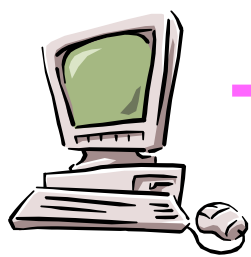
- ⌚ Informação adicional no URL;

- ⌚ Dados enviados no corpo de uma mensagem.

**CGI**



# O Conceito de *gateway programs*



1



**HTTP  
Server**

1. O Cliente envia os dados para o servidor:

@ A *query string* como parte do URL;

@ Informação adicional no URL;

@ Dados enviados no corpo de uma mensagem.

Por exemplo:

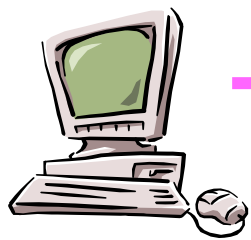
```
http://krillin/cgi-cajo/ola?NOME=Carlos&IDADE=29
```

A informação **NOME=Carlos&IDADE=29** actua como *query string* que é passada ao servidor. Quando o servidor recebe os dados, constrói uma variável de ambiente com esta informação e de seguida cria uma instância do programa **ola**.

Por sua vez o programa CGI(**ola**) pode obter os dados por análise da variável ambiente **QUERY\_STRING**.



# O Conceito de *gateway programs*



1



**HTTP  
Server**

1. O Cliente envia os dados para o servidor:

@ A *query string* como parte do URL;

@ Informação adicional no URL;

**CGI**

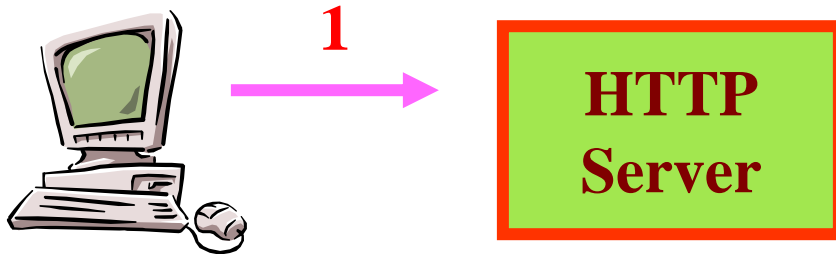
Esta informação adicional é referente a *path*. Mais uma vez é passada ao programa CGI através de variáveis ambientes. Um exemplo disso pode ser:

```
http://krillin/cgi-cajo/ola/dir/file?NOME=Carlos+Jorge&IDADE=29
```

dos enviados no  
mensagem.



# O Conceito de *gateway programs*



1.O Cliente envia os dados para o servidor:

⌚ A *query string* como parte do URL;

⌚ Informação adicional no URL;

⌚ Dados enviados no corpo de uma mensagem.

Em alternativa a passar a informação através do URL, os dados podem ser passados ao programa CGI no corpo de uma mensagem. Este método é usado tipicamente no tratamento de *forms* HTML. Uma das vantagens face à passagem dos dados usando o URL, é que, a dimensão dos dados não fica limitada à dimensão das variáveis ambientes.



# O Conceito de *gateway programs*



**HTTP  
Server**



**CGI**

2. O servidor envia os dados para o programa CGI:

- ⌚ Argumentos na linha de comando;
- ⌚ Variáveis de ambiente;
- ⌚ Através do *standard input*.



# O Conceito de *gateway programs*



**HTTP  
Server**

2

Os dados são passados ao programa CGI como argumentos na linha de comando. Este mecanismo só é usado quando o pedido teve origem com o método GET usando uma *query* do tipo ISINDEX.

2. O servidor envia os dados para o programa CGI:

- @ Argumentos na linha de comando;
- @ Variáveis de ambiente;
- @ Através do *standard input*.





# O Conceito de *gateway programs*



**HTTP  
Server**

Os dados a passar ao programa CGI são colocados em variáveis ambiente antes de o programa CGI se executar. É esta a forma utilizada caso seja recebido um comando HTTP **GET**.

Além dos dados que foram enviados pelo browser, são também criadas variáveis de ambiente com informação adicional, tais como:

- directoria raiz do servidor WEB;
- método utilizado no pedido (`REQUEST_METHOD`);
- dados concatenados no URL (`QUERY_STRING`);
- tipo de servidor (`SERVER_SOFTWARE`);
- nome do domínio;
- etc.

2. O servidor envia os dados para o programa CGI:

- @ Argumentos na linha de comando;
- @ Variáveis de ambiente;
- @ Através do *standard input*.



# O Conceito de *gateway programs*



HTTP  
Server

2

O programa CGI pode ler os dados do *standard input*. É deste modo que os programas CGI conseguem obter os dados que foram enviados pelos clientes usando o método **POST**.

Os servidores criam sempre uma variável de ambiente que indica se os dados foram passados usando o método GET ou POST (`REQUEST_METHOD`);.

2. O servidor envia os dados para o programa CGI:

- @ Argumentos na linha de comando;
- @ Variáveis de ambiente;
- @ Através do *standard input*.



# O Conceito de *gateway programs*



3. O CGI envia os dados para o servidor através do seu *standard output*:
- ⌚ Com processamento posterior por parte do servidor Web;
  - ⌚ Sem processamento posterior por parte do servidor Web;



# O Conceito de *gateway programs*



**HTTP  
Server**



O programa CGI devolve o resultado ao servidor por escrita dos dados no *standard output*. De um modo geral a informação produzida por um CGI é composta por duas partes:

1. Conjunto de directivas para o servidor. Estas directivas ajudam o servidor a compor a informação que vai ser enviada ao cliente (*Cabeçalhos HTTP*).
  2. Os dados que representam o pedido que o utilizador efectuou.
- Estas duas partes são separadas por uma linha composta por CRLF (*carriage return linefeed*).

3. O CGI envia os dados para o servidor através do seu *standard output*:

- ⌚ Com processamento posterior por parte do servidor Web;
- ⌚ Sem processamento posterior por parte do servidor Web;



# O Conceito de *gateway programs*



**HTTP  
Server**

3

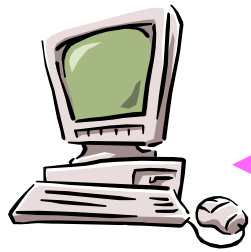
Existe uma excepção ao método descrito em (1). Se o nome do programa CGI começa por **nph** (*non-parsed header*) o servidor não efectua nenhum pré-processamento sobre os dados e envia-os directamente para o *browser*. É da responsabilidade do programa CGI providenciar a totalidade do cabeçalho da resposta HTTP.

3. O CGI envia os dados para o servidor através do seu *standard output*:

- ⌚ Com processamento posterior por parte do servidor Web;
- ⌚ Sem processamento posterior por parte do servidor Web;



# O Conceito de *gateway programs*



4

**HTTP  
Server**

4. O servidor devolve os dados ao cliente que vê o seu pedido satisfeito.

**CGI**



# CGI - Exemplo

## Exemplo de um CGI em BASH

**Data.cgi**

```
#!/bin/bash
echo "Content-type: text/html"
echo
echo "<html> <head> <title> Ola - Exemplo de um CGI </title>
    </head>"
echo "<body> <h1> Exemplo de um CGI</h1>"
echo "<p>"
date
echo "</p>"
echo "</body> </html>"
```





# Tecnologias para Geração de Páginas Dinamicamente

- ❏ CGI (*Common Gateway Interface*);

- ❏ FastCGI;

- ❏ Server Extensions APIs;

  - ❏ ISAPI (*Internet Server API*);

- ❏ SSI (*Server Side Includes*);

  - ❏ SSJS (*Server Side JavaScript*);

  - ❏ ASP (*Active Server Pages*);

  - ❏ JSP (*Java Server Pages*);

- ❏ Servlets;





# Geração de Páginas Dinamicamente

Tecnologia	Suporte em várias linguagens	Portável	Interagem com o servidor em tempo de execução
CGI	✓	✓ / X	X
FastCGI	✓	✓ / X	X
ASP	✓	X	
JSP	X (Java)	JVM	✓
ISAPI	✓	X	✓
SSJS	X (JavaScript)	✓	
Servlets	X (Java)	JVM	✓



# Formulários HTML

( forma de recolher dados para enviar ao servidor )




# Formulários HTML


- É uma das opções disponíveis no HTML para fazer a recolha de dados de modo a serem enviados para o servidor;
- Permite especificar se queremos enviar os dados através do método HTTP **GET** ou **POST**;
  - **GET** – Os dados são concatenados no fim do URL no pedido **HTTP**.
  - **POST** – Os dados são inseridos no corpo da mensagem do pedido **HTTP**.



# HTML – Formulários

 Elemento que permite a introdução de dados, pelo utilizador, no browser e envia para o servidor HTTP (ex: inserir palavras num motor de procura, responder a um inquérito)

 **<form> ... </form>** – Delimita um formulário

 **Action** – Atributo que indica o *URL para onde enviar os dados (CGI, Servlet, jsp, asp, ...)*

 **Method** – Atributo que especifica o método de envio dos dados

- ⦿ Method= “post” – Indica que é para ser enviada uma mensagem HTTP do tipo *POST com os dados no corpo da mensagem*
- ⦿ Method= “get” Indica que é para ser enviada uma mensagem HTTP do tipo *GET com os dados concatenados no URL*

 **Target** – Indica a frame onde será visualizada a resposta

- ⦿ Pode ser uma frame específica referindo o seu nome (ex: target=“myframe1”)
- ⦿ Pode ser uma frame relativa
  - ⦿ Target=“\_blank” – Nova janela
  - ⦿ Target=“\_self” – Na própria frame
  - ⦿ Target=“\_parent” – No frameset pai
  - ⦿ Target=“\_top” – No corpo da própria janela



# HTML – Formulários (cont)

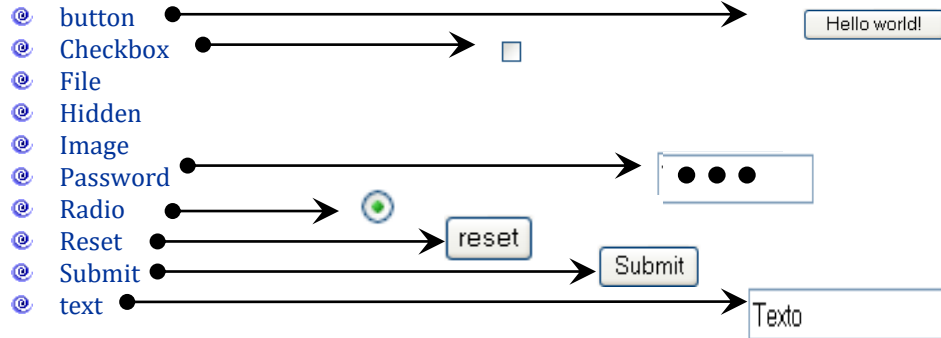
 Dentro de um formulário podemos ter os elementos:

 `<input>` – Cria uma entrada no formulário.

⌚ Atributos – accept, align, alt, checked, disabled, maxlength, **name**, size, readonly, src, **type**, **value**

⌚ Não tem etiqueta de fim

⌚ Tipos possíveis para o atributo type:



 `<select> ... </select>` – Permite especificar uma *drop down list*

⌚ Os seus atributos são:

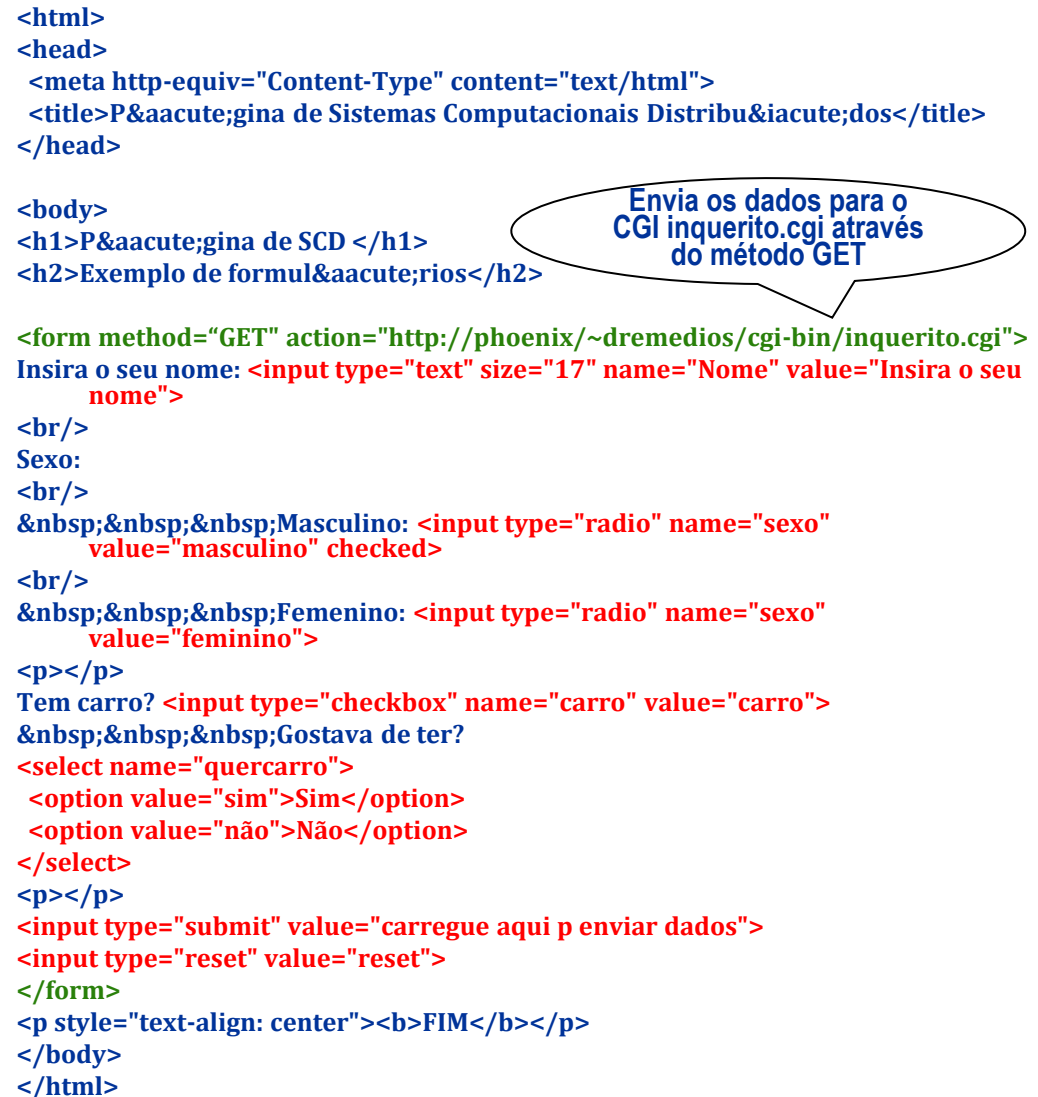
- ⌚ `disabled`
- ⌚ `multiple`
- ⌚ `name`
- ⌚ `size`

⌚ `<option>` – introduz uma opção na lista, e os seus atributos são:

- ⌚ `disabled`
- ⌚ `label`
- ⌚ `selected`
- ⌚ `value`

 `<textarea> ... </textarea>` – Área em que pode ser introduzido texto (Ex: comentários)

- ⌚ `name`
- ⌚ `Cols`
- ⌚ `Rows`





# CGI – Exemplo “ShellScript”

Código fonte,  
ficheiro  
inquerito.cgi

```
#!/bin/bash

echo "Content-type: text/html"
echo
echo "<html>"

#testar qual o metodo de envio dos dado
if [ $REQUEST_METHOD = "GET" ]
then
    # GET - Ler dados da QUERY_STRING

    DADOS=$QUERY_STRING
else
    #post - Ler dados do stdin
    #get input from stdin
    read DADOS
fi

echo "<HEAD>"
echo "<meta http-equiv=\"Content-Type\" \"
    content=\"text/html\">"

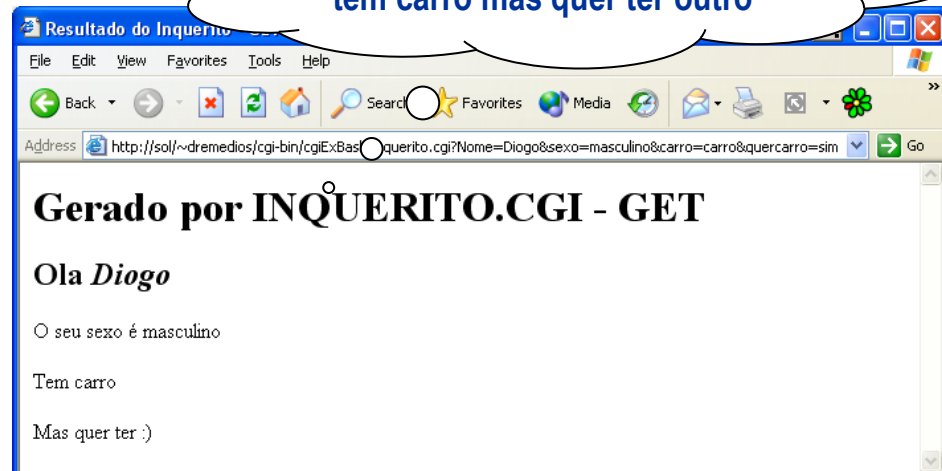
# Vamos gerar o resto da pagina
if [ $REQUEST_METHOD = "GET" ]
then
    # GET
    echo "<TITLE>Resultado do Inquerito - GET</TITLE>"
    echo "</HEAD>"
    echo "<BODY>"
    echo "<H1>Gerado por INQUERITO.CGI - GET</H1>"
else
    #post
    echo "<TITLE>Resultado do Inquerito - POST</TITLE>"
    echo "</HEAD>"
    echo "<BODY>"
    echo "<H1>Gerado por INQUERITO.CGI - POST</H1>"
fi
```

```
# fazer o parsing dos DADOS

# iterar os varios parametros
IFS='&'
for param in $DADOS; do
# echo "param: $param\n"
case $param in
    Nome=*)      echo "<H2>Ola <i>`echo $param | cut -c6-
                20`</i></H2>";;
    sexo=*)      echo "<p>O seu sexo &eacute; `echo $param |
                cut -c6-20`</p>" ;;
    carro=carro) echo "<p>Tem carro</p>" ;;
    quercarro=sim) echo "<p>Mas quer ter :)</p>" ;;
    quercarro=nao) echo "<p>Nem quer ter :P</p>" ;;
esac

done
echo "</body></html>"
```

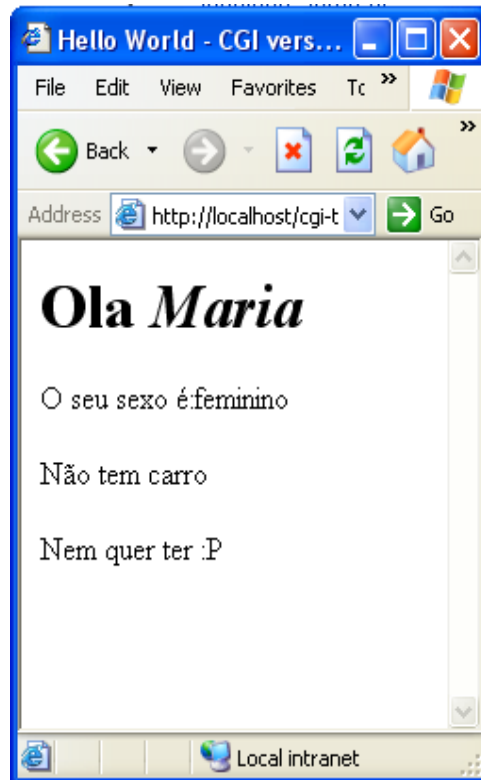
Escolhendo no formulário do  
slide anterior o Nome: Diogo,  
tem carro mas quer ter outro





# CGI – Exemplo “C”

Escolhendo no formulário o  
Nome: Maria, não tem carro mas  
também não quer ter



```
#include <stdlib.h>
#include <time.h>
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include "../Include/html-lib.h"
#include "../Include/cgi-lib.h"
```

```
int main() {
    char saudacao[64];
    llist entries;
```

```
    read_cgi_input(&entries);
    html_header();
    html_begin("Hello World - CGI version");
```

```
    char * userName = cgi_val(entries, "Nome");
    char * sexo = cgi_val(entries, "sexo");
    char * carro = cgi_val(entries, "carro");
    char * quercarro = cgi_val(entries, "quercarro");
```

```
    sprintf(saudacao, "Ola <i>%s</i>", userName);
    h1(saudacao);
    printf("<p>O seu sexo é:%s</p>", sexo);
    if(carro==NULL){
        printf("Não tem carro");
        if(!strcmp(quercarro,"sim"))
            printf("<p>Mas quer ter :)</p>");
        else
            printf("<p>Nem quer ter :P</p>");
    }else
        printf("<p>Tem carro</p>");
    html_end();
```

```
    return 1;
}
```

Código fonte,  
ficheiro  
inquerito.c

O CGI  
inquerito.cgi é o  
ficheiro binário  
que resulta de  
compilar  
inquerito.c





# CGI - Conclusões

- ❏ Os programas CGI podem ser realizados recorrendo a qualquer tipo de linguagem de programação ou de *scripting* (*BASH*, *C shell*, *C*, *Perl*, *Tcl*).
- ❏ O resultado destes programas pode ser virtualmente em qualquer formato (texto, HTML, POSTSCRIPT, PDF, imagens, etc)
  - ❏ Desde que o cliente tenha capacidade para o tratar.
  - ❏ O programa CGI pode saber quais os tipos que pode retornar consultando o cabeçalho HTTP "Accept:..."

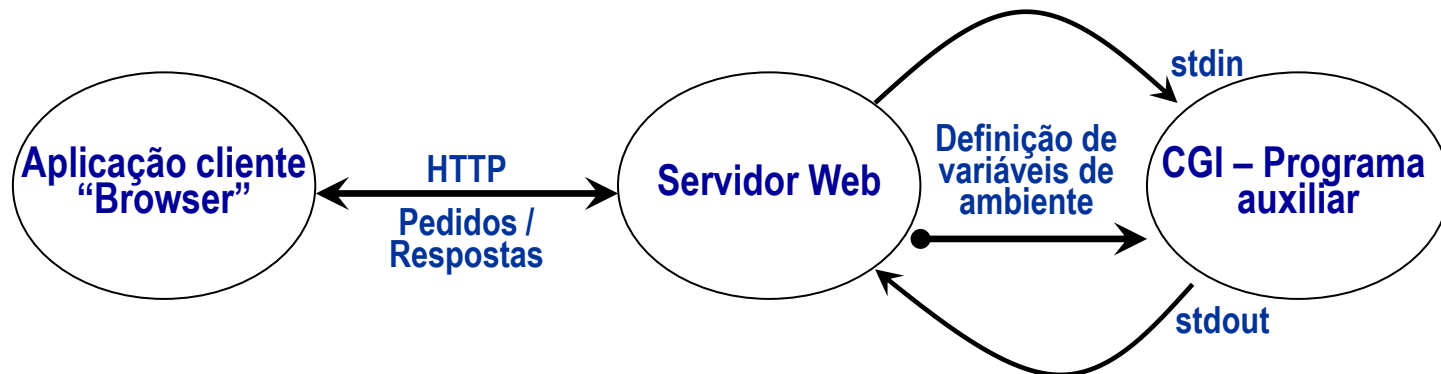


# CGI - Conclusões



## Common Gateway Interface (CGI)

- Define um interface entre o servidor Web e programas que se executem na mesma máquina;
- Caso o pedido ao servidor Web referencie um programa e este tenha permissões para ser executado, este dá início à sua execução;
- A comunicação entre o servidor Web e o programa auxiliar é feita através dos mecanismos oferecidos pelo sistema operativo (stdin, stdout e as variáveis de ambiente);
- O output gerado pelo programa auxiliar é devolvido pelo servidor web para o browser, normalmente corresponde a uma página HTML;





# “*Bibliografia*” utilizada

- RFC's:
  - RFC 1945 – HTTP 1.0
  - RFC 2616 – HTTP 1.1
  
- HTML 4.0 Sourcebook
  - Graham, Ian S. – John Wiley & Sons; 4th edition
  
- CGI Programming Unleashed;
  - *Eugene Eric Kim*
  
- Folhas da cadeira
  - Criação e Utilização de CGI's
    - ⊗ *Nuno Oliveira, Carlos Gonçalves*
  
- Especificação CGI
  - <http://hoohoo.ncsa.uiuc.edu/cgi/overview.html>
  - <http://hoohoo.ncsa.uiuc.edu/cgi/examples.html>





# Glossário de Termos

 WWW	– World-Wide Web;
 HTML	– HyperText Markup Language;
 HTTP	– HyperText Transfer Protocol;
 URL	– Uniform Resource Locator;
 URI	– Uniform Resource Identifier;
 URN	– Uniform Resource Name;
 RFC	– Request For Comments;
 CGI	– Common Gateway Interface;
 API	– Application Programming Interface;
 ASP	– Active Server Pages;
 JSP	– JavaServer Pages;